Training - Entrenamiento

Objetivo General

- Establecer conexión con las plataformas bases y el proyecto de entrenamiento y realizar efectivamente el entrenamiento.

Objetivos especificos

- Armar la infraestructura (Modelo, API's) para conectarnos correctamente con las plataformas base (Saturno, Nutresa, Latam, etc) y con el proyecto de entrenamiento (Training-Automate).
- Construir las apis necesarias para que los pasos del entrenamiento se lleven a cabo
- Determinar que plataforma enciende cualquier instancia de máquina.

Actividades

El módulo de entrenamiento va a realizar las acciones que anteriormente realizaban los sistemas *Saturno*, *Nutresa* o *Latam* al momento de entrenar imagenes, de ahora en adelante estos sistemas mencionados se van a comunicar con integrations brindándonos los parámetros y nosotros vamos por decirlo de alguna manera a descargar a esos sistemas de ciertas acciones como en este caso el proceso de entrenar imagenes.

Actividades requeridas

- Migración de los modelos(13), ordenados en app's(6) dentro del app **core**, revisar la sección ***base de datos*** de la `carpeta documentacion/doc_tec_rest_manager.md`, ahí se detalla como se debe declarar el modelo.
- Importar la lógica que se tenia en el py y adaptarla al proyecto de integrations, básicamente cambiar de `APIView` a `ModelViewSet`.

Funciones

- **change_status_for_all_instance_of_machines**: Esta función es la encargada de replicar la información del cambio de estado de una InstanceMachine en todas las Bases de datos, admite como parámetro el status: int y unos **kwargs que es el medio a través del cual se filtra para llegar al modelo.
- **get_database**: Esta función es la encargada de extraer el nombre del modelo StarMachine, recibe como parámetro uuid para poder obtener el nombre de la base de datos de la relación propia del modelo.
- **StartMachineSave**: Esta función es la encargada de guardar en el modelo StartMachine el momento en el cual una maquina es encendida, esta se colocá en status_machine=True, al finalizar un entrenamiento esta instancia se coloca en status_machine=False, indicando que la misma esta apagada.

API's o ViewSet's

- **InitTrainingViewSet**: Esta api recibe la orden desde las plataformas para encender la máquina y ella envia los parametros a un celery task **turn_on_machine** y ocupa esa máquina en todas las bases de dato de las demás plataformas, anulando así que esta pueda ser utilizada por cualquier otra plataforma en ese momento.
- **ReceiverInstanceViewSet**: Una vez la máquina enciende, esta api recibe los parametros de ip, nombre y quien encendió dicha maquina y envia los parametros de las imagenes que se van a entrenar y las caracteristicas de costumbre (por supuesto que también va quién encendió esta máquina).
- **FinishTrainingViewSet**: Esta api recibe ciertos parametros que nos ayudan a determinar quién encendio la máquina el status de la misma un msg, el transaction_id del modelo de entrenamiento entre otros lo cual nos permite a nosotros asignar estados, apagar la máquina y volver a colocar las mismas disponibles para futuros entrenamientos.

Flujo

- Las plataforma principales nos va solicitar mediante un `token de acceso`, (este ya estará en el archivo de *enviroments*) un segundo token de autenticación (este lo respondemos nosotros a través del api obtain token).
- Teniendo el token de autenticación nos va poder pasar los parámetros a una primera api init training, esta va a enviar a través de una tarea *Celery* llamada **turn_on_machine** a que la máquina sea encendida.
    - **turn_on_machine** es una tarea celery que en lineas generales da la orden para encender la instancia de la maquina para iniciar el entrenamiento, en esta se valida la existencia del **ModelTraining** y de **InstanceMachine**, luego de esto se asignan msj y estados (éxito o error) según sea el destacar que para este proceso entra en juego una nueva función **StartMachineSave** que se encarga de almacenar en un nuevo modelo quién enciende (plataforma) la máquina remota, esto es importante en el flujo pues con esto también sabemos a cual base de datos apuntar y queda constancia del flujo de entrada y salida de cada proceso.
- Ya con la máquina encendida entonces esta va enviarnos a *integrations* una señal indicandonos su ip, el nombre de la instancia y ahora el transaction_id de quién enciende la máquina a la api de integrations receiver instance, la cual se va encargar de determinar con ese token a cual base de datos pertenece esa máquina y nos va enviar a una tarea *Celery* llamada **training_start**, la cual va ser la encargada de retornarle los parámetros a ***training-automate***, este es el encargado de ***ENTRENAR*** el conjunto de imagenes.
    - **training_start**, esta tarea se encarga de validar el modelo **ModelTraining** y a partir de este extrae los parámetros que posteriormente vamos a enviar al proyecto ***training-automate*** que es quien da inicio al entrenamiento.
- Luego de ciertos procesos en ***training-automate*** este va enviarnos a *integrations* una señal con ciertos parámetros a la api finish training, para apagar la instancia de la máquina y desactivar el registro del modelo **StartMachine** que se encuentra entrenando y asignar los valores correspondiente a los estados del modelo de entrenamiento y otros modelos relacionados en la base de datos adecuada y replicando la información en el resto de las bases de dato.